MA 3046 - Matrix Analysis

Problem Set 3 - Practical Considerations in Numerical Linear Algebra

1. Consider a notional computer which uses a decimal-based ($\beta = 10$), three-digit floating-point number system. (Neglect the allowable range of exponents.) Assume the system has a temporary double-length accumulator, which then rounds or chops, as appropriate, all intermediate results back to three significant digits before performing subsequent calculations. Compute, under both rounding and chopping, the following:

a. $12.3 + .0234$
b. $-.0321 + .000136$
c. $12.3 - .0234$
d. $-321 + 32.1$
e. $132 * 0.987$
f. $-2.14/.000137$
g. $(-.111 + .222) * (.00111)/999$

2. Assume the same notional computer as used in problem 1.

a. Simulate, for $x = 1.09$, chopping arithmetic, the calculation of the expression:

$$2.93x^3 - 2.74x^2 + 3.06x - 4.77$$

where your calculation proceeds from left to right, and integer powers are computed by repeated multiplication (e.g. $x^3 = (x * x) * x$).

b. Repeat part a. using simulated rounding arithmetic.

c. Determine both the actual absolute and absolute relative errors in your answers to parts a and b. Are these relative errors reasonable for this machine?

3. According to basic trigonometric theory, $\sin(n\pi) = 0$ for **every** integer n. Consider the following MATLAB code:

```
nexp = 0:16 ;
y = sin( (10.^nexp)*pi ) ;
plot( nexp, y )
```

Run this code, and explain the resulting graph.

4. a. Describe, as completely as possible, what is meant by the terms *cache*, *RAM* and *swap*, in the context of (desktop) computers.

b. Distinguish between the terms *serial processor*, *parallel processor* and *pipeline processor* in the context of computers.

5. a. Approximately how many flops will be required to compute the matrix product

$$\mathbf{A}\,\mathbf{B}\,\mathbf{C}$$

where $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{B} \in \mathbb{R}^{n\times p}$, and $\mathbf{C} \in \mathbb{R}^{p\times r}$?

b. Approximately how many flops will be required for each of the following, theoretically identically, computations, assuming $\mathbf{q}$ and $\mathbf{u}$ are both $n \times 1$, real column vectors:

(i.) $\left(\mathbf{I} - \mathbf{q}\,\mathbf{q}^T\right)\mathbf{u}$

(ii.) $\mathbf{u} - \left(\mathbf{q}\mathbf{q}^T\right)\mathbf{u}$

(iii.) $\mathbf{u} - \left(\mathbf{q}^T\mathbf{u}\right)\,\mathbf{q}$

c. Are there any reasons, besides flops count, why one of the computations in part b. above might be preferable?

6. a. Consider the computation

$$\mathbf{A} = \mathbf{I} - \mathbf{u}\,\mathbf{v}^H$$

where both $\mathbf{u}$ and $\mathbf{v}$ are general 2000×1 column vectors. Identify what is inefficient about each of the following MATLAB codes

(i.) **for i=1:2000 ; for j=1:2000 ; a(i,j) = 1 - u(i)*v(j) ; end ; end**

(ii.) **for i=1:2000 ; a(i , :) = 1 - u(i)*v′ ; end**

b. What would be a highly efficient MATLAB implementation of the computation in part a? What would be the flop count for that implementation, compared the the flop count of the two implementations above?

c. How would you change your code in part b. if you knew the first 1600 elements of both $\mathbf{u}$ and $\mathbf{v}$ were identically zero? How would the flop count of your new code compare with the flop count of your code for part a.?